

图文 105 NameServer是如何处理Broker的注册请求的？

159 人次阅读 2020-02-26 10:42:22

[详情](#) [评论](#)

NameServer是如何处理Broker的注册请求的？



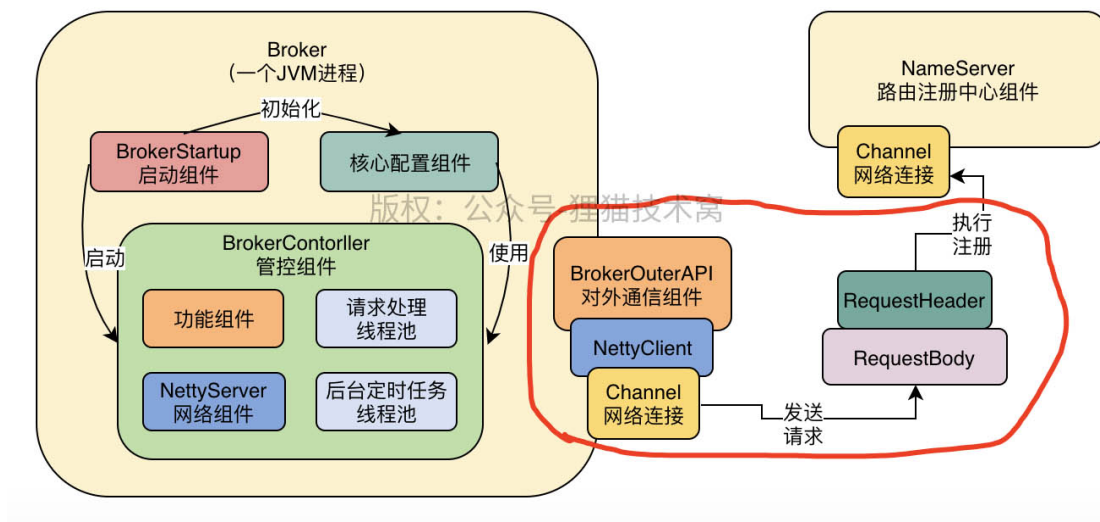
继《从零开始带你成为JVM实战高手》后，阿里资深技术专家携新作再度出山，重磅推荐：

(点击下方蓝字试听)

[《从零开始带你成为MySQL实战优化高手》](#)

上一次我们分析完了Broker启动的时候是如何通过BrokerOuterAPI发送注册请求到NameServer去的

大家看下图红圈的部分，我们可以回忆一下这个Broker发送注册请求的过程。



今天我们就来研究一下NameServer接收到这个注册请求之后，是如何进行处理的，这里要涉及到Netty网络通信相关的东西，可能很多人没接触过Netty，但是没关系，我尽量弱化掉Netty自身的东西，主要站在通用的网络通信的角度去讲解。

大家如果对Netty不了解的，对一些Netty自己的特殊API也不用过多的去关注，主要了解他的网络通信的流程就可以了。

现在我们回到NamesrvController这个类的初始化的方法里去，也就是NamesrvController.initialize()这个方法

我们看下面的一个源码片段就可以了，我省略了一些无关紧要的代码。

```

1 public boolean initialize() {
2
3     this.kvConfigManager.load();
4
5     // 在这里初始化了Netty服务器，这个大家应该都没问题了
6     this.remotingServer = new NettyRemotingServer(
7         this.nettyServerConfig, this.brokerHousekeepingService);
8
9     this.remotingExecutor =
10         Executors.newFixedThreadPool(nettyServerConfig.getServerWorkerThreads(), new ThreadFactoryImpl("RemotingExecutorThread_"));
11
12     // 非常核心的是下面这行代码，他有一个注册Processor的过程
13     // 这个Processor其实就是请求处理器，是NameServer用来处理网络请求的组件
14     this.registerProcessor();
15
16     // 省略了一大堆源码
17
18 }

```

我们继续看下面的registerProcessor()方法的源码。

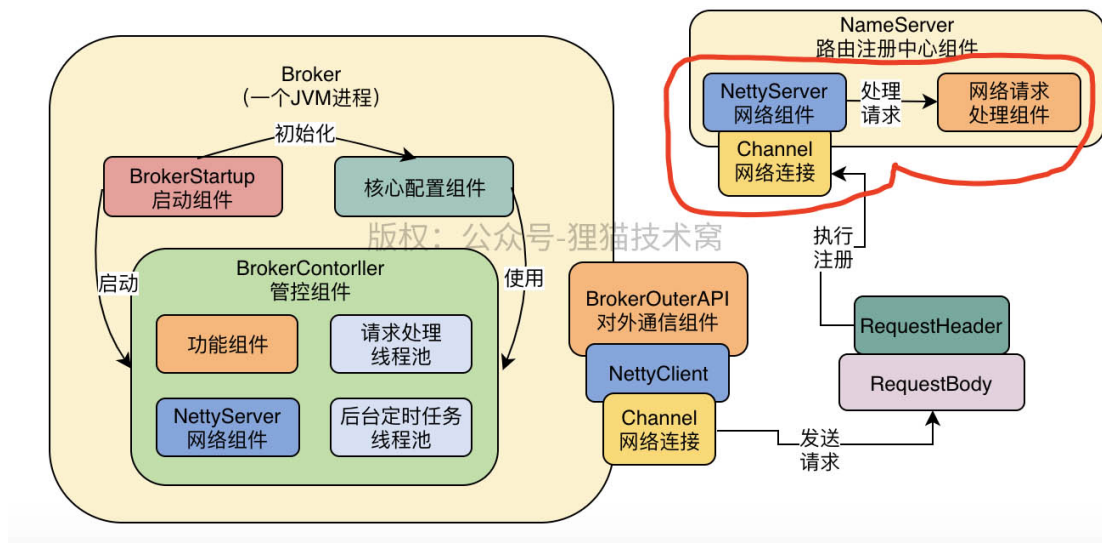
```

1 private void registerProcessor() {
2     if (namesrvConfig.isClusterTest()) {
3
4         this.remotingServer.registerDefaultProcessor(new ClusterTestRequestProcessor(this, namesrvConfig.getProductEnvName()),
5             this.remotingExecutor);
6     } else {
7         // 上面那个代码是用于处理测试集群的，核心是下面这行代码
8         // 大家可以看到，在这里，把NameServer的默认请求处理组件注册了进去
9         // 是注册给了NettyServer的
10        // 也就是说，NettyServer接收到的网络请求，都会由这个组件来处理
11        this.remotingServer.registerDefaultProcessor(new DefaultRequestProcessor(this), this.remotingExecutor);
12    }
13 }
14

```

大家看完了上面的源码之后，我来给大家在图里感受一下，在下图中我们可以看到NettyServer是用于接收网络请求的，那么接收到的网络请求给谁处理呢？

其实就是给DefaultRequestProcessor这个请求处理组件来进行处理的。



所以我们如果要知道Broker注册请求是如何处理的，直接就是看DefaultRequestProcessor中的代码就可以了，下面给大家看一下这个类的一些源码片段。

```
1 @Override
2 public RemotingCommand processRequest(
3     ChannelHandlerContext ctx,
4     RemotingCommand request) throws RemotingCommandException {
5
6     // 下面这坨代码就不用管他了，打印调试日志的
7     if (ctx != null) {
8         log.debug("receive request, {} {} {}",
9             request.getCode(),
10             RemotingHelper.parseChannelRemoteAddr(ctx.channel()),
11             request);
12     }
13
14     // 下面就是核心代码了，根据你的请求类型有不同的处理过程
15     switch (request.getCode()) {
16         case RequestCode.PUT_KV_CONFIG:
17             return this.putKVConfig(ctx, request);
18         case RequestCode.GET_KV_CONFIG:
19             return this.getKVConfig(ctx, request);
20         case RequestCode.DELETE_KV_CONFIG:
21             return this.deleteKVConfig(ctx, request);
22         case RequestCode.QUERY_DATA_VERSION:
23             return queryBrokerTopicConfig(ctx, request);
24         // 别的请求先不用管了，直接看下面这个好了，就是注册Broker的请求
25         case RequestCode.REGISTER_BROKER:
26             Version brokerVersion = MQVersion.value2Version(request.getVersion());
27             if (brokerVersion.ordinal() >= MQVersion.Version.V3_0_11.ordinal()) {
28                 return this.registerBrokerWithFilterServer(ctx, request);
29             } else {
30                 // 核心的注册请求处理逻辑，就是在下面的registerBroker方法里
31                 return this.registerBroker(ctx, request);
32             }
33         // 省略了一大堆源码
34     }
35 }
```

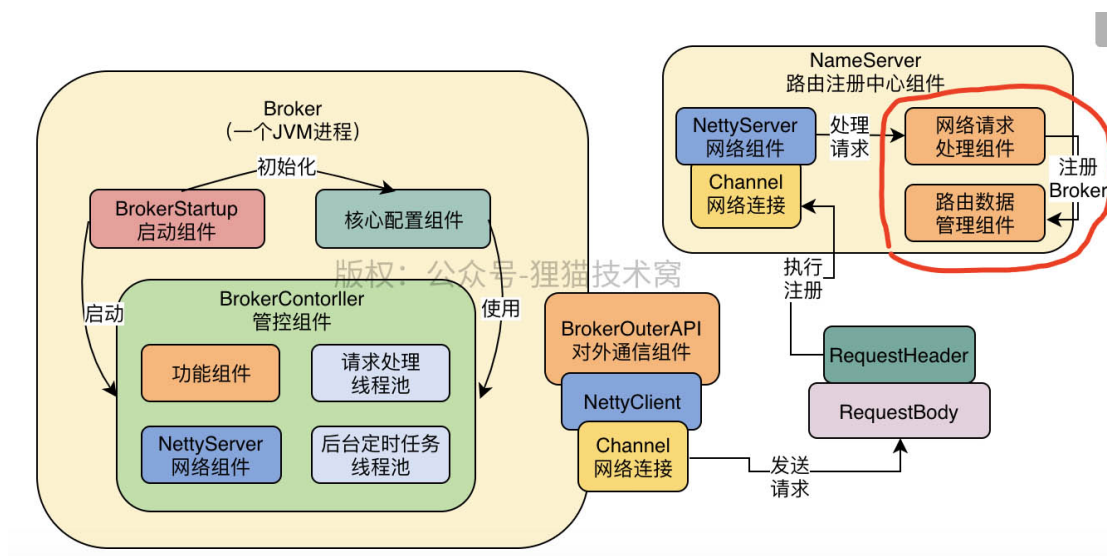
接着我们进入这个类里的registerBroker()方法，去看看到底如何完成Broker注册的。

```

1 public RemotingCommand registerBroker(
2     ChannelHandlerContext ctx,
3     RemotingCommand request) throws RemotingCommandException {
4     final RemotingCommand response = RemotingCommand.createResponseCommand(RegisterBrokerResponseHeader.class);
5
6     // 下面一大坨代码大致看一眼就行了，就是解析注册请求，然后构造返回响应的
7     final RegisterBrokerResponseHeader responseHeader = (RegisterBrokerResponseHeader) response.readCustomHeader();
8     final RegisterBrokerRequestHeader requestHeader =
9         (RegisterBrokerRequestHeader) request.decodeCommandCustomHeader(RegisterBrokerRequestHeader.class);
10
11     if (!checksum(ctx, request, requestHeader)) {
12         response.setCode(ResponseCode.SYSTEM_ERROR);
13         response.setRemark("crc32 not match");
14         return response;
15     }
16
17     TopicConfigSerializeWrapper topicConfigWrapper;
18     if (request.getBody() != null) {
19         topicConfigWrapper = TopicConfigSerializeWrapper.decode(request.getBody(), TopicConfigSerializeWrapper.class);
20     } else {
21         topicConfigWrapper = new TopicConfigSerializeWrapper();
22         topicConfigWrapper.getDataVersion().setCounter(new AtomicLong(0));
23         topicConfigWrapper.getDataVersion().setTimestamp(0);
24     }
25
26     // 核心其实在这里，就是调用了RouteInfoManager这个核心功能组件
27     // RouteInfoManager，顾名思义，就是路由信息管理组件，他是一个功能组件
28     // 调用了这个功能组件的注册Broker的方法
29     RegisterBrokerResult result = this.namesrvController.getRouteInfoManager().registerBroker(
30         requestHeader.getClusterName(),
31         requestHeader.getBrokerAddr(),
32         requestHeader.getBrokerName(),
33         requestHeader.getBrokerId(),
34         requestHeader.getHaServerAddr(),
35         topicConfigWrapper,
36         null,
37         ctx.channel()
38     );
39
40     // 下面都是在构造返回的响应了，大致看一眼就行了
41     responseHeader.setHaServerAddr(result.getHaServerAddr());
42     responseHeader.setMasterAddr(result.getMasterAddr());
43
44     byte[] jsonValue = this.namesrvController.getKvConfigManager().getKVListByNamespace(NamespaceUtil.NAMESPACE_ORDER_TOPIC_CONFIG);
45     response.setBody(jsonValue);
46     response.setCode(ResponseCode.SUCCESS);
47     response.setRemark(null);
48     return response;
49 }

```

下面我们先在图里给大家体现一下RouteInfoManager这个路由数据管理组件，实际Broker注册就是通过他来做的。



至于RouteInfoManager的注册Broker的方法，我们就不带着大家来看了。这里给大家留一个今天的源码分析小作业，大家可以自己到RouteInfoManager的注册Broker的方法里去看看，最终如何把一个Broker机器的数据放入RouteInfoManager中维护的路由数据表里去的。

其实我这里提示一下，核心思路非常简单，无非就是用一些Map类的数据结构，去存放你的Broker的路由数据就可以了，包括了Broker的clusterName、brokerId、brokerName这些核心数据。

而且在更新的时候，一定会基于Java并发包下的ReadWriteLock进行读写锁加锁，因为在这里更新那么多的内存Map数据结构，必须要加一个写锁，此时只能有一个线程来更新他们才行！

大家看完这个Broker注册的最后一个步骤之后，也就是RouteInfoManager的注册过程之后，有什么源码分析的心得体会，都可以在评论区发表出来，跟大家一起交流。

End

专栏版权归公众号**狸猫技术窝**所有

未经许可不得传播，如有侵权将追究法律责任

狸猫技术窝精品专栏及课程推荐：

[《从零开始带你成为JVM实战高手》](#)
[《21天互联网Java进阶面试训练营》（分布式篇）](#)
[《互联网Java工程师面试突击》（第1季）](#)
[《互联网Java工程师面试突击》（第3季）](#)

重要说明：

如何提问：每篇文章都有评论区，大家可以尽情留言提问，我会逐一答疑

如何加群：购买狸猫技术窝专栏的小伙伴都可以加入狸猫技术交流群，一个非常纯粹的技术交流的地方

具体加群方式，请参见目录菜单下的文档：《付费用户如何加群》（**购买后可见**）